

Python: module vcs.vector

vcs.vector

[index](#)

Vector (Gv) module

Modules

[vcs.Canvas](#)

[vcs.VCS validation functions](#)

[vcs. vcs](#)

[cdtime](#)

[vcs.queries](#)

[vcs](#)

Classes

[builtin .object](#)

[Gv](#)

class **Gv**([builtin .object](#))

Class: [Gv](#)

Vector

Description of [Gv](#) Class:

The vector graphics method displays a vector plot of a 2D vector field. Vectors are located at the coordinate locations and point in the direction of the vector field. Vector magnitudes are the product of data vector field and a scaling factor. The example below shows how to modify the vector alignment, type, and reference.

This class is used to define an vector table entry used in VCS, or used to change some or all of the vector attributes in an existing entry.

Other Useful Functions:

a=vcs.init()	# Constructor
a.show('vector')	# Show predefined vector graphics
a.show('line')	# Show predefined VCS line objects
a.setcolormap("AMIP")	# Change the VCS color Map
a.vector(s1, s2, v, 'default')	# Plot data 's1', and 's2' with and 'default' template
a.update()	# Updates the VCS Canvas at once
a.mode=1, or 0	# If 1, then automatic update 0, then use update function to update the VCS Canvas.

Example of Use:

a=vcs.init()

To Create a new instance of vector use:

```
vc=a.createvector('new','quick')      # Copies content of 'quick'  
vc=a.createvector('new')             # Copies content of 'default'
```

To Modify an existing vector use:

```
vc=a.getvector('AMIP_psl')
```

```
vc.list()                           # Will list all the vector a  
vc.projection='linear'            # Can only be 'linear'  
lon30={-180:'180W', -150:'150W', 0:'Eq'}  
vc.xticlabels1=lon30  
vc.xticlabels2=lon30  
vc.xticlabels(lon30, lon30)        # Will set them both  
vc.xmtics1=''  
vc.xmtics2=''  
vc.xmtics(lon30, lon30)           # Will set them both  
vc.yticlabels1=lat10  
vc.yticlabels2=lat10  
vc.yticlabels(lat10, lat10)        # Will set them both  
vc.ymtics1=''  
vc.ymtics2=''  
vc.ymtics(lat10, lat10)           # Will set them both  
vc.datawc_y1=-90.0  
vc.datawc_y2=90.0  
vc.datawc_x1=-180.0  
vc.datawc_x2=180.0  
vc.datawc(-90, 90, -180, 180)      # Will set them all  
xaxisconvert='linear'  
yaxisconvert='linear'  
vc.xyscale('linear', 'area_wt')    # Will set them both
```

Specify the line style:

```
vc.line=0                           # Same as vc.line='solid'  
vc.line=1                           # Same as vc.line='dash'  
vc.line=2                           # Same as vc.line='dot'  
vc.line=3                           # Same as vc.line='dash-dot'  
vc.line=4                           # Same as vc.line='long-dot'
```

Specify the line color of the vectors:

```
vc.linecolor=16                     # Color range: 16 to 230, de  
vc.linewidth=1                      # Width range: 1 to 100, def
```

Specify the vector scale factor:

```
vc.scale=2.0                        # Can be an integer or float
```

Specify the vector alignment:

```
vc.alignment=0                      # Same as vc.alignment='head'  
vc.alignment=1                      # Same as vc.alignment='cent  
vc.alignment=2                      # Same as vc.alignment='tail'
```

Specify the vector type:

```
vc.type=0                           # Same as vc.type='arrow head'
```

```

vc.type=1                      # Same as vc.type='wind barbs'
vc.type=2                      # Same as vc.type='solid arrows'

Specify the vector reference:
vc.reference=4                  # Can be an integer or float

```

Methods defined here:

__init__(self, parent, Gv_name=None, Gv_name_src='default', createGv=0)

datawc(self, dsp1=1e+20, dsp2=1e+20, dsp3=1e+20, dsp4=1e+20)

list(self)

rename = renameGv(self, old_name, new_name)

```

#####
#
# Function:      renameGv
#
# Description of Function:
#           Private function that renames the name of an existing
#           graphics method.
#
#
# Example of Use:
#           renameGv(old_name, new_name)
#           where: old_name is the current name of vector
#                   new_name is the new name for the vector
#
#####

```

script(self, script_filename=None, mode=None)

Function: script # Calls _vcs.scr

Description of Function:

Saves out a vector graphics method in Python or VCS script
a designated file.

Example of Use:

script(scriptfile_name, mode)

where: scriptfile_name is the output name of the
mode is either "w" for replace or "a" for append.

Note: If the filename has a ".py" at the end
Python script. If the filename has a ".scr"
produce a VCS script. If neither extension
is present or if the mode is "a" then by default
a Python script will be produced.

```

a=vcs.init()
vec=a.createboxfill('temp')
vec.script('filename.py')
```

Append to a Python file

```

    vec.script('filename.scr')           # Append to a VCS file
    vec.script('filename', 'w')

xmtics(self, xmt1=", xmt2=")

xticlabels(self, xtl1=", xtl2=")

xyscale(self, xat=", yat=")

ymtics(self, ymt1=", ymt2=")

yticlabels(self, ytl1=", ytl2=")

```

Properties defined here:

alignment

```

get">get = _getalignment(self)
set">set = _setalignment(self, value)

```

datawc_calendar

```

get">get = _getcalendar(self)
set">set = _setcalendar(self, value)

```

datawc_timeunits

```

get">get = _gettimeunits(self)
set">set = _settimeunits(self, value)

```

datawc_x1

```

get">get = _getdatawc_x1(self)
set">set = _setdatawc_x1(self, value)

```

datawc_x2

```

get">get = _getdatawc_x2(self)
set">set = _setdatawc_x2(self, value)

```

datawc_y1

```

get">get = _getdatawc_y1(self)
set">set = _setdatawc_y1(self, value)

```

datawc_y2

```

get">get = _getdatawc_y2(self)
set">set = _setdatawc_y2(self, value)

```

level

```

get">get = _getlevels(self)
set">set = _setlevels(self, value)

```

levels

```

get">get = _getlevels(self)
set">set = _setlevels(self, value)

```

line

get">**get** = _getline(self)
set">**set** = _setline(self, value)

linecolor

get">**get** = _getlinecolor(self)
set">**set** = _setlinecolor(self, value)

linewidth

get">**get** = _getlinewidth(self)
set">**set** = _setlinewidth(self, value)

name

get">**get** = _getname(self)
set">**set** = _setname(self, value)

projection

get">**get** = _getprojection(self)
set">**set** = _setprojection(self, value)

reference

get">**get** = _getreference(self)
set">**set** = _setreference(self, value)

scale

get">**get** = _getscale(self)
set">**set** = _setscale(self, value)

type

get">**get** = _gettype(self)
set">**set** = _settype(self, value)

xaxisconvert

get">**get** = _getxaxisconvert(self)
set">**set** = _setxaxisconvert(self, value)

xmtics1

get">**get** = _getxmtics1(self)
set">**set** = _setxmtics1(self, value)

xmtics2

get">**get** = _getxmtics2(self)
set">**set** = _setxmtics2(self, value)

xticlabels1

get">**get** = _getxticlabels1(self)
set">**set** = _setxticlabels1(self, value)

xticlabels2

get">**get** = _getxticlabels2(self)
set">**set** = _setxticlabels2(self, value)

yaxisconvert

```

get">get = _getyaxisconvert(self)
set">set = _setyaxisconvert(self, value)

ymtics1
get">get = _getymtics1(self)
set">set = _setymtics1(self, value)

ymtics2
get">get = _getymtics2(self)
set">set = _setymtics2(self, value)

yticlabels1
get">get = _getyticlabels1(self)
set">set = _setyticlabels1(self, value)

yticlabels2
get">get = _getyticlabels2(self)
set">set = _setyticlabels2(self, value)

```

Data and other attributes defined here:

```

__slots__ = ['setmember', 'parent', 'name', 'g_name', 'xaxisconvert', 'yaxisconvert', 'linecolor', 'line',
'xticlabels2', 'yticlabels1', 'yticlabels2', 'xmtics1', 'xmtics2', 'ymtics1', 'ymtics2', 'datawc_x1', 'datawc_y1']

g_name = <member 'g_name' of 'Gv' objects>

parent = <member 'parent' of 'Gv' objects>

setmember = <member 'setmember' of 'Gv' objects>

```

Functions

```

getGvmember(self, member)
#####
#
# Function:      getGvmember
#
# Description of Function:
#      Private function that retrieves the vector members from the
#      structure and passes it back to Python.
#
#
# Example of Use:
#      return_value =
#      getGvmember(self, name)
#      where: self is the class (e.g., Gv)
#              name is the name of the member that is being
#
#####

```

```

getmember = getGvmember(self, member)
#####
#
# Function:      getGvmember
#
# Description of Function:
#      Private function that retrieves the vector members from the
#      structure and passes it back to Python.
#
#
# Example of Use:
#      return_value =
#      getGvmember(self, name)
#              where: self is the class (e.g., Gv)
#                      name is the name of the member that is being
#
#####
renameGv(self, old_name, new_name)
#####
#
# Function:      renameGv
#
# Description of Function:
#      Private function that renames the name of an existing vector
#      graphics method.
#
#
# Example of Use:
#      renameGv(old_name, new_name)
#              where: old_name is the current name of vector graph
#                      new_name is the new name for the vector graph
#
#####
setGvmember(self, member, value)
#####
#
# Function:      setGvmember
#
# Description of Function:
#      Private function to update the VCS canvas plot. If the value
#      is set to 0, then this function does nothing.
#
#
# Example of Use:
#      setGvmember(self, name, value)
#              where: self is the class (e.g., Gv)
#                      name is the name of the member that is being
#                      value is the new value of the member (or att
#
#####

```

```
setmember = setGvmember(self, member, value)
#####
#
# Function:      setGvmember
#
# Description of Function:
#      Private function to update the VCS canvas plot. If the can
#      set to 0, then this function does nothing.
#
#
# Example of Use:
#      setGvmember(self,name,value)
#      where: self is the class (e.g., Gv)
#              name is the name of the member that is being
#              value is the new value of the member (or att
#
#####
#####
```

Data

StringTypes = (<type 'str'>, <type 'unicode'>)